Coding Challenge : WhatsApp-Based Food Ordering System

Intern Coding Challenge: WhatsApp-Based Food Ordering System

Objective

You are required to build a **Food Ordering System** that allows users to browse menus, place orders, and receive updates via **WhatsApp messaging**. Your system will consist of:

- A **FastAPI backend** that:
  - Manages menu items, orders, and delivery status.
  - Integrates with **WhatsApp API** (Twilio API, Meta's WhatsApp Business API) to send and receive messages.
  - Sends real-time **order status updates** to customers.
- A **ReactJS frontend** that:
  - Provides a dashboard to view and manage orders.
  - Allows restaurant staff to update order statuses.
- A **Python SDK** (generated using OpenAPI Generator CLI) for programmatic access to the system.
- **Automation scripts** (PowerShell, Shell, or any other script) to simplify setup and execution.

Your solution should demonstrate **best practices** in:

- **FastAPI integration** with external services (WhatsApp API).
- **React UI flows** for managing food orders.
- **OpenAPI documentation** for SDK generation.
- **Automated scripts** for quickly deploying the system.

Leverage **LLMs, open-source libraries, or API documentation** where applicable. Creativity in adding extra features is encouraged.

Tasks & Requirements

1. Backend Development (FastAPI + WhatsApp API)

Menu Management

- **POST** `/menu/`
- **Request Body:** Item Name, Description, Price, Availability Status
- **Behavior:**
  - Validate and store the item in the database.
  - Allow enabling/disabling availability.
- **Response:** Menu item details with ID.
- **GET** `/menu/` → Retrieve all menu items.
- **GET** `/menu/{item_id}` → Retrieve details of a specific menu item.

Order Placement via WhatsApp

- **POST** `/orders/`
- **Request Body:** Customer Name, WhatsApp Number, List of Menu Items
- **Behavior:**
  - Check if the requested items are available.

  - Create an order and send a **WhatsApp order confirmation message** to the user.
- **Response:** Order details with status confirmation.

Order Status Updates

- **PATCH** `/orders/{order_id}`
- **Request Body:** New Status (`pending`, `preparing`, `out-for-delivery`, `delivered`)
- **Behavior:**
  - Update the order status in the database.
  - Send a **WhatsApp message** with the updated status.
- **Response:** Confirmation message.

Retrieve Orders

- **GET** `/orders/` → List all active orders.
- **GET** `/orders/{order_id}` → Retrieve details of a specific order.

Cancel Order

- **DELETE** `/orders/{order_id}`
- **Behavior:**
  - Mark the order as canceled.
  - Send a **WhatsApp cancellation message** to the customer.
- **Response:** HTTP 204 or JSON confirmation message.

OpenAPI Docs

- Ensure **FastAPI** exposes an OpenAPI spec (`http://localhost:8000/openapi.json`).
- Document request/response schemas properly.

Unit Tests

- Tests for:
  - Menu management, order creation, and cancellation.
  - Ensuring WhatsApp notifications are sent.
  - Handling invalid requests.

---

2. Frontend Client (ReactJS)

Develop a ReactJS Dashboard that Communicates with FastAPI

- **Menu Management**
  - Form to add/edit menu items.
  - Display current menu with availability status.
- **Order Management**
  - View all active orders and statuses.
  - Update order status (`preparing`, `out-for-delivery`, `delivered`).
- **UI/UX Considerations**
  - Focus on **ease of use & error handling**.
  - Display **order success/failure** messages.

---

3. Python SDK (OpenAPI Generator CLI)

Generate the SDK

- Use the OpenAPI spec (`http://localhost:8000/openapi.json`).
- Example command:

```
openapi-generator-cli generate -i http://localhost:8000/openapi.json -g python -o whatsapp_food_sdk

```

Validate & Use the SDK

- After generation, ensure it supports:
  - **add_menu_item()** → Add a new menu item.
  - **place_order()** → Place a food order.
  - **update_order_status()** → Update the order status.
  - **list_orders() / get_order_by_id()** → Retrieve order information.

Sample Script for SDK Usage

```
from whatsapp_food_sdk.api.orders_api import OrdersApi
from whatsapp_food_sdk import ApiClient

client = ApiClient()
orders_api = OrdersApi(client)

# Retrieve all orders
orders = orders_api.get_orders()
print(orders)

```

---

4. Automation Scripts

Setup Script (PowerShell, Bash, etc.)

- **Python Virtual Environment**
  - Create & activate a virtual environment.
- **Install Python Dependencies**

```
pip install -r requirements.txt

```

(Should include `fastapi, uvicorn, twilio (or equivalent), celery`, etc.)

- **Configure WhatsApp API Credentials**
  - Store API keys in **.env** or a secure credentials file.
  - Document how to obtain and configure them.
- **Install React Dependencies**

```
npm install

```

Execution Script

- **Start FastAPI Backend**

```
uvicorn main:app --host 0.0.0.0 --port 8000
```

```

```

- **Start React Frontend**

```
npm start

```

Completion Criteria

- **Functional System:**
  - Allows users to **browse menu, place orders, and receive updates**.
  - Sends **WhatsApp confirmation & order status messages**.
  - Provides a **ReactJS dashboard** for management.
- **WhatsApp Integration:**
  - Uses **Twilio API, WhatsApp Business API**, or an equivalent service.
  - Handles **real-time messaging updates**.
- **Python SDK:**
  - **Generated via OpenAPI.**
  - **Demonstrated with a sample script.**
- **Automation:**
  - One script to set everything up.
  - One script (or set of commands) to run the system.
- **Testing:**
  - Backend tests covering order creation, updates, and WhatsApp notifications.

Bonus Features (Optional)

- **Two-Way WhatsApp Interaction**
  - Allow customers to reply with "Cancel" to cancel an order.
- **Estimated Delivery Time**
  - Provide a real-time estimated delivery time for each order.
- **Loyalty System**
  - Track repeat customers and offer discounts via WhatsApp messages.
- **Multi-Restaurant Support**
  - Allow orders from multiple restaurants.
- **Detailed Error Messages**
  - Provide clear feedback for **unavailable menu items, invalid phone numbers, or API errors**.

Deliverables

- **Backend (FastAPI) source code**
- **ReactJS frontend code**
- **Python SDK (OpenAPI generated)**
- **Setup & Execution Scripts**
- **Unit tests for backend**
- **README with setup instructions**